SWE30011 – THE INTERNET OF THINGS PROGRAMMING

Individual Assignment (Practical)

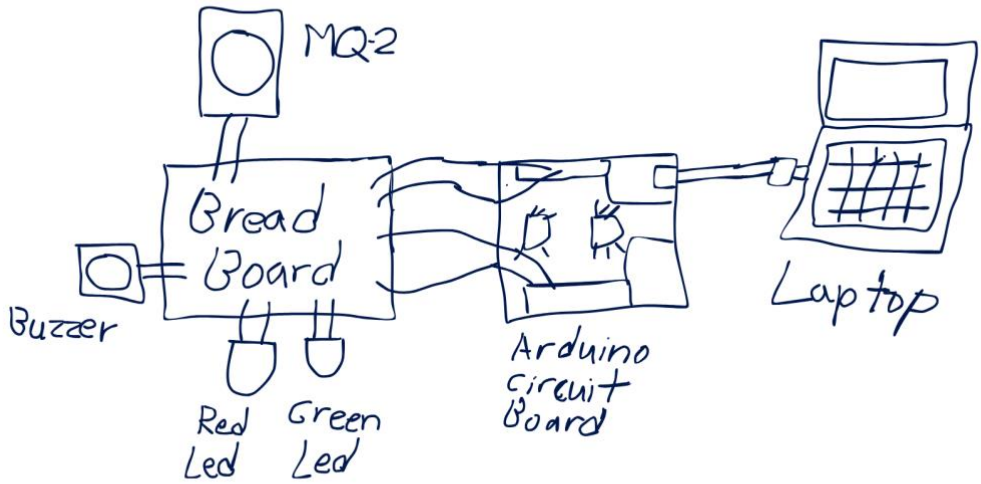
KEATH KOR
103844324

Introduction

In recent years, the severity and frequency of fire incidents have been increasing globally. This result in a devasting effect that can cause huge risks to live and the environment. Cause of fire incidents have been on the rise, especially with the recent bush fires happening in Australia, it is important to address and understand these issues. This is crucial for developing measures to mitigate risks and impacts of future fire incidents. In this project, a tool is created to reduce these impacts using the implementation of The Internet of Things (IOT). If the device detects fire or smoke, it will immediately sound an alarm, alerting residents to take necessary actions to stop or reduce the spread of the fire.

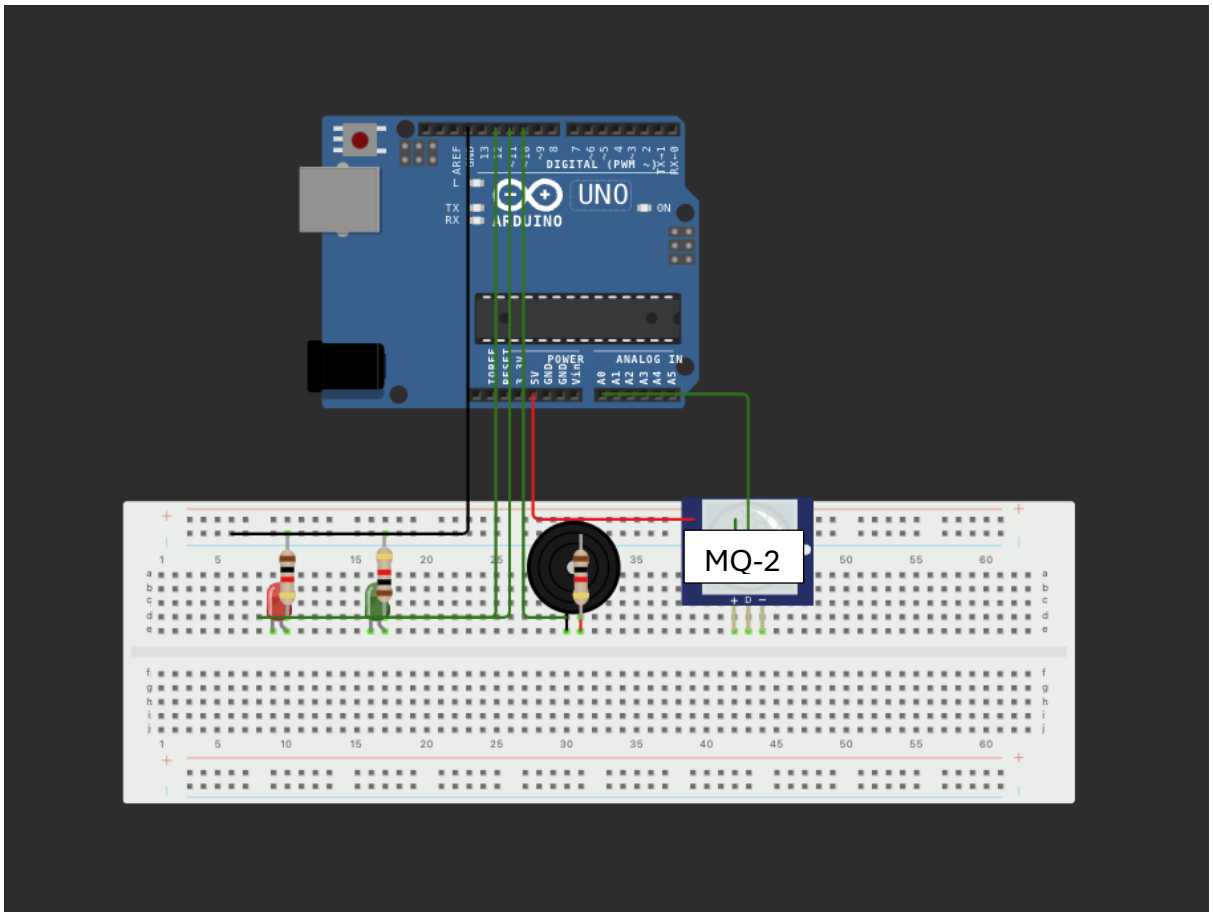
Pyroward is built and used for this project to make an Arduino Smoke or Fire detector Circuit. The circuit consists of an MQ2-Sensor, 1 red led light pin, 1 green led light pin, a buzzer, 3 transistors, and 8 jumper cables. When fire or smoke is detected by the sensor, it will send an alert to a channel notifying the users to take immediate action. Utilising this tool, house fires or fire incidents can be prevented, resulting in a safer and more secured environment and communities.

Conceptual design

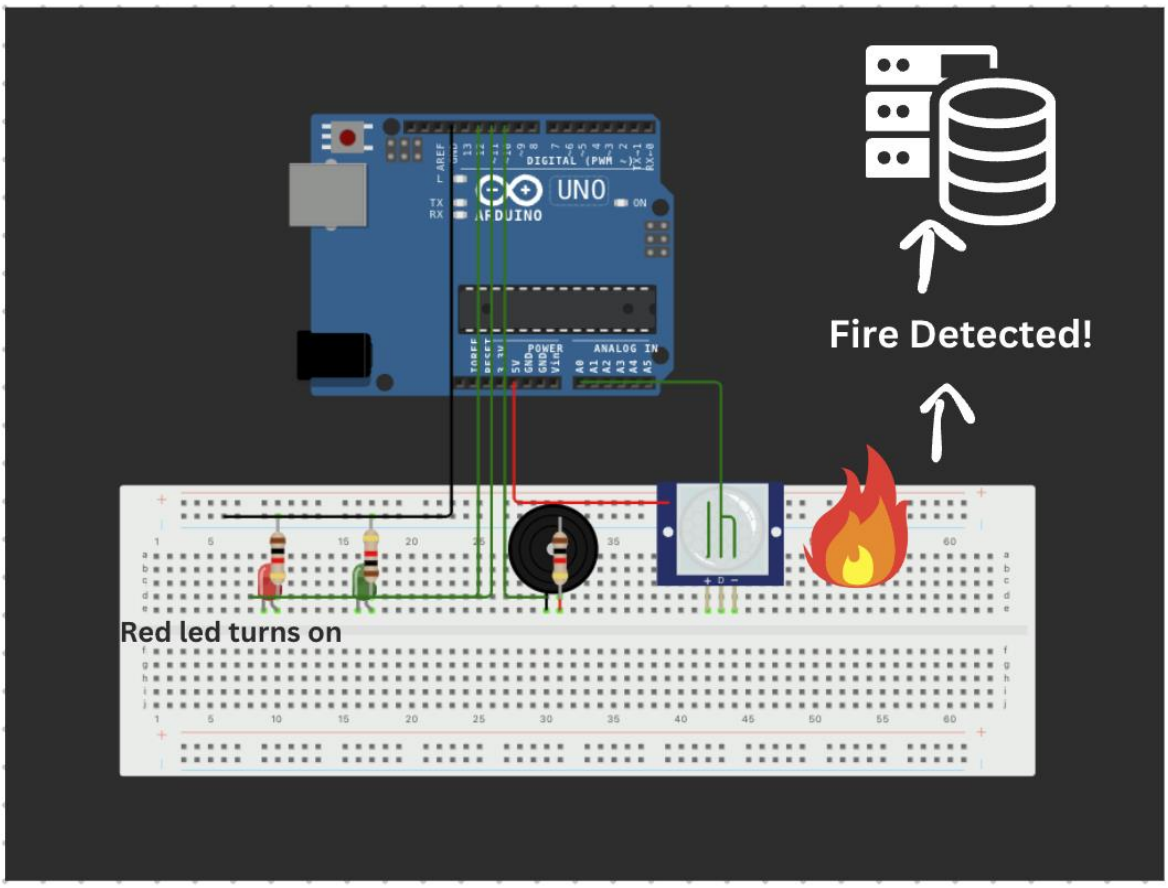


Implementation

The components for this IOT device consist of: Arduino UNO, MQ-2 sensor, Red Led, Green Led, and a buzzer, connecting to a laptop or computer for power.

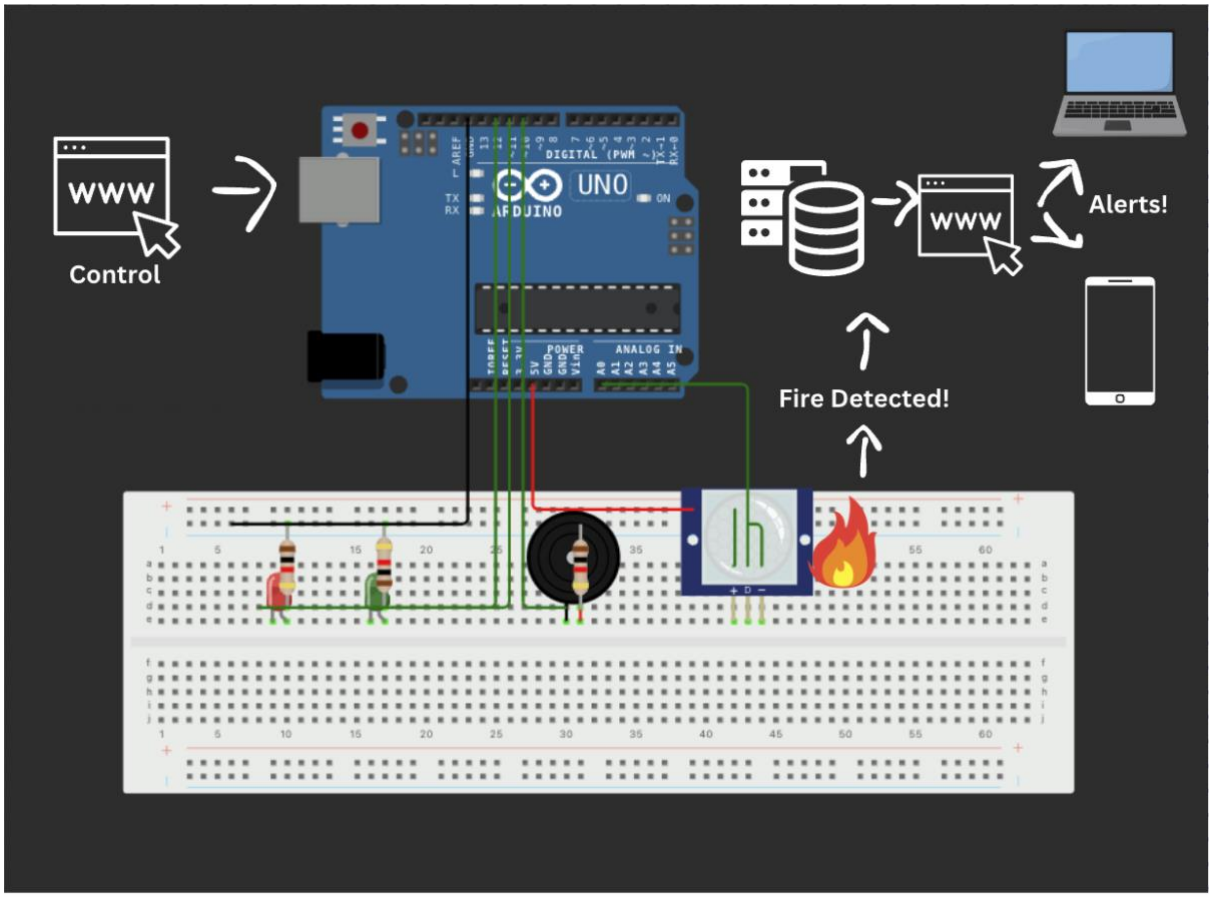


To begin with, the digital data can be read out of the MQ2 sensor, connected to the Arduino digital pin A0 (PIN A0). The green led is connected to pin 11 (PIN 11), to indicate that Pyroguard is on. The red led is connected to pin 12 (PIN 12), to alert and turn on when smoke is detected. The buzzer is connected to pin 10 (PIN 10), buzzing whenever fire or smoke is detected over the threshold >300. When the system is on, the green led is turned to indicate that Pyroward is on and active. When fire is detected, the buzzer goes off, the green led turns off and the red led turns on, indicating fire or smoke detected.



In the next step, results are generated and can be viewed via the serial monitor. If fire is detected, alert messages will be sent to the Edge server also via a message channel. The results are recorded onto a fire database implemented via MySQL. Devices that are subscribed to the channel will receive alerts such as: "**`Fire Detected! Take action immediately.`**"

The smoke detector records the smoke input going to pin AO, once fire or smoke input reaches over 300, the red led turns on and buzzer goes off, and fire detected alert messages will pop up. Alert messages are then sent to devices informing them about the fire.



Finally, the Arduino's smoke sensor's records can be viewed online via a web interface. The Arduino board can be controlled on the webserver, such as turning on and off the led and buzzer.

How to run:

Using a Unix terminal system as an alternative to a virtual machine, the system can be run by:

To run Pyroward and record results to the database:

- Set up virtual environment via terminal
- source my_venv/bin/activate
- cd ~/Desktop
- cd fireproject
- python3 iot.py

To run Pyroward with the whole interface with message system:
source my_venv/bin/activate
cd ~/Desktop
cd fireproject
python controlsys.py

Appendix

Pyroward Arudino IDE:

Arduino Uno

Smokedetector.ino

```
// Assigning pin numbers for components
int greenLed = 11;   // Pin number for the green LED
int redLed = 12;     // Pin number for the red LED
int smokeA0 = A0;    // Analog pin number for the smoke sensor
int buzzer = 10;     // Pin number for the buzzer

// Threshold value for smoke detection
int sensorThres = 300;

void setup() {
  // Setting up pin modes
  pinMode(redLed, OUTPUT);    // Sets the red LED pin as an output
  pinMode(greenLed, OUTPUT);  // Sets the green LED pin as an output
  pinMode(buzzer, OUTPUT);    // Sets the buzzer pin as an output
  pinMode(smokeA0, INPUT);    // Sets the smoke sensor pin as an input
  Serial.begin(9600);         // Initializes serial communication at 9600 baud rate
}

void loop() {
  // Reading analog value from the smoke sensor
  int analogSensor = analogRead(smokeA0);

  // Printing analog sensor value to the serial monitor
  Serial.print("Pin A0: ");
  Serial.println(analogSensor);
  delay(1000);  // Delay for 1 second

  // Checking if the analog sensor value exceeds the threshold
  if (analogSensor > sensorThres) {
    // Smoke or fire detected
    digitalWrite(redLed, HIGH);  // Turn on the red LED
    digitalWrite(greenLed, LOW); // Turn off the green LED
    tone(buzzer, 1000, 200);      // Generate a tone on the buzzer
    Serial.println("Fire Detected!");  // Print message to the serial monitor
  } else {
    // No smoke or fire detected
    digitalWrite(redLed, LOW);   // Turn off the red LED
    digitalWrite(greenLed, HIGH);  // Turn on the green LED
    noTone(buzzer);               // Stop the buzzer tone
  }
  delay(100);  // Delay for 100 milliseconds before the next iteration
}
```

Iot.py to run recording to database

iot.py 3 ×

iot.py > ...

```python
import serial
import mysql.connector
from notify_run import Notify
from flask import Flask


notify = Notify()
device = '/dev/cu.usbmodem1101'

# Update the port path with the correct one for your Arduino
ser = serial.Serial(device, 9600, timeout=1)  # Set timeout to 1 second

# Connect to MySQL database
db_conn = mysql.connector.connect(host='localhost', user='root', password='cutie123', database='fire_db')

# Create cursor object
cursor = db_conn.cursor()

try:
    while True:
        # Read data from Arduino until timeout
        data = ser.readline().decode().strip()  # Read one line of data
        if data:
            print(data)

            # Insert data into the database
            insert_query = "INSERT INTO fire_logs (fire_detected) VALUES (%s)"
            cursor.execute(insert_query, (data,))
            db_conn.commit()

        if "fire" in data.lower():
            print("Fire detected! Sending Alerts!")
            notify.send("Fire Detected! Take action immediately.")

except KeyboardInterrupt:
    print("Stopping data acquisition.")

finally:
    # Close cursor and database connection
    cursor.close()
    db_conn.close()
    ser.close()

```

Controlsys.py to run webserver, data recording along with alerts channel.

controlsys.py 3

controlsys.py > ...

```python
from flask import Flask, render_template
import mysql.connector
import serial
from notify_run import Notify
import threading
import subprocess

app = Flask(__name__)

pins = {
    11: {'name': 'Green LED', 'state': 0},
    12: {'name': 'Red LED', 'state': 0},
    'A0': {'name': 'Smoke Sensor', 'state': 0},
    10: {'name': 'Buzzer', 'state': 0}
}

db_conn = mysql.connector.connect(host='localhost', user='root', password='cutie123', database='fire_db')
cursor = db_conn.cursor()

notify = Notify(endpoint='https://notify.run/c/FJuU08XX8dyd8UzRKKCF/')

@app.route('/')
def index():
    # Fetch fire logs from the database
    cursor.execute("SELECT * FROM fire_logs")
    fire_logs = cursor.fetchall()

    # Prepare HTML to display fire logs
    fire_logs_html = "<h1>Fire Logs</h1>"
    fire_logs_html += "<ul>"
    for log in fire_logs:
        fire_logs_html += f"<li>{log[1]}</li>"
    fire_logs_html += "</ul>"

    templateData = {
        'pins': pins,
        'fire_logs': fire_logs_html
    }
    return render_template('index.html', **templateData)

@app.route("/<changePin>/<toggle>")
def toggle_function(changePin, toggle):
    changePin = int(changePin)
    deviceName = pins[changePin]['name']
    ser = serial.Serial('/dev/cu.usbmodem1101', 9600, timeout=1)
```

```python
    ser.flush()

    if toggle == "on":
        ser.write(str(changePin).encode())
        pins[changePin]['state'] = 1
        message = "Turned " + deviceName + " on."
    elif toggle == "off":
        ser.write(str(changePin + 10).encode())
        pins[changePin]['state'] = 0
        message = "Turned " + deviceName + " off."

    templateData = {
        'pins': pins
    }
    return render_template('index.html', **templateData)

@app.route('/fire-alert', methods=['POST'])
def fire_alert():
    # Handle the incoming alert
    # You can perform any necessary actions here, such as sending notifications
    print("Fire alert received!")
    return "Fire alert received!", 200

def read_arduino_data():
    notify = Notify()
    device = '/dev/cu.usbmodem1101'

    # Update the port path with the correct one for your Arduino
    ser = serial.Serial(device, 9600, timeout=1)  # Set timeout to 1 second

    try:
        while True:
            # Read data from Arduino until timeout
            data = ser.readline().decode().strip()  # Read one line of data
            if data:
                print(data)

            if "fire" in data.lower():
                print("Fire detected! Sending Alerts!")
                subprocess.run(["curl", "https://notify.run/FJuUO8XX8dyd8UzRKKCF", "-d", "Fire Detected! Take action immediately."])

    except KeyboardInterrupt:
        print("Stopping data acquisition.")
```

```python

    finally:
        # Close serial connection
        ser.close()

if __name__ == '__main__':
    # Start Flask app without the reloader
    flask_thread = threading.Thread(target=app.run, kwargs={'debug': True, 'host': '0.0.0.0', 'port': 8080, 'use_reloader': False})
    flask_thread.start()

    # Start Arduino data reading loop in the main thread
    read_arduino_data()
```

Index.html for web interface

index.html

templates > index.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>Arduino Web Server</title>
</head>
<body>
    <h1>Arduino Web Server</h1>
    <h2>Toggle Buttons</h2>
    {% for pin, details in pins.items() %}
        <h3>{{ details.name }} is currently {% if details.state == 1 %}<strong>on</strong>{% else %}<strong>off</strong>{% endif %}</h3>
        <div class="row">
            <div class="col-md-2">
                {% if details.state == 1 %}
                    <a href="/{{ pin }}/off" class="btn btn-block btn-lg btn-default" role="button">Turn off</a>
                {% else %}
                    <a href="/{{ pin }}/on" class="btn btn-block btn-lg btn-primary" role="button">Turn on</a>
                {% endif %}
            </div>
        </div>
    {% endfor %}

    <h2>Fire Logs</h2>
    {{ fire_logs | safe }}

    </div>
</body>
</html>
```

Resources

[How does work MQ2 sensor – MQ2 sensor with Arduino UNO [Code and circuit diagram]]

[How To Use Gas Sensor MQ-2 With Arduino]

[Smoke Detector using Gas Sensor]

[Message channel]